

FRC Project*Libre*_{TM}¹ Tutorial

Using Project*Libre* for Your Robot Build

Version 2

Craig Yankes
Western & Central PA *FIRST* Regional Planning Committee
cyankes@msa.com

¹ Project*Libre* is a trademark of Marc O'Brien and Laurent Chretienneau.

Contents

Introduction	3
General Comments	3
Scope of this Tutorial	3
Resources for more Information or Help.....	3
Getting Started.....	4
Creating a Project.....	4
Normal Display	4
Navigating in this Tutorial	5
Defining Your Project	5
Adding Tasks	5
Adding Dependencies	6
Setting Work Days.....	9
Using Critical Paths	10
Tracking the Project	11
Task Completion Percentage	11
Adjusting Late Tasks.....	12
Wrap-up	14
Deferred Topics.....	14
Creating a Project with Forward versus Backwards Scheduled Tasks	14
New Topics in the Second Version of the Tutorial.....	16
Context for Version One	16
Resources.....	16
Appendix 1 (from V1): 2013 Team 3511 Task List	32

Introduction

General Comments

Congratulations! Reading this shows that you are interested in using project management software to help with your team's robot build. While using this type of software to help manage a 6-week project might seem excessive, I believe you will find that this will help you deal with the hectic build period by quickly showing where your team is in the build process and whether it is behind, on, or ahead of schedule. Quite importantly, if your team is behind schedule a tool like this can help identify where the team needs to concentrate its effort to achieve the goal of taking a quality robot to Regionals and, possibly, further on to Nationals.

Two comments before we get into the “meat” of the subject. First, in writing this tutorial I am presuming that you have either attended the “Project Planning” session at the 2013 SCRA² Workshop or have read the annotated slides from that session that are posted with this document. The slides cover the introductory material explaining why a team can benefit from using project-planning techniques, introduces some of the terminology that is used and describes some common pitfalls that hopefully you will be able to avoid by being aware of them. While the terminology will be defined in this tutorial, the “whys” and “oops” in the slides will not be recreated here. If you haven't seen them yet, I suggest looking over the slides before continuing with this tutorial.

The second comment is one of thanks. In the spring 2013 robot build, Team 3511 used an Excel³ spreadsheet to view their progress and word of that spreadsheet led to the Project Planning session being requested at the SCRA Workshop. Jim Broker, Team 3511's mentor, has given permission for that spreadsheet to be used in this tutorial as example tasks or as templates as desired. Thank you, Jim!

Scope of this Tutorial

It is impossible in a short tutorial to introduce you to every aspect of project management software whether represented in a free tool like *ProjectLibre* or commercial tools like Microsoft Project⁴. Therefore, not only is this not going to be a comprehensive user's manual describing every option and feature in *ProjectLibre*, the scope of this tutorial is being purposely limited to the smallest set of features you'll likely need to manage the robot build. In short, the goal of this tutorial is not to impress you with the tool, but rather to help you use this tool in your build. If you are intrigued by this software, there is plenty of time after the build and competitions are over to explore it more thoroughly.

Resources for more Information or Help

There are several resources available to help you learn and use the tool. There is an online help facility in the tool accessible by clicking on the “?” in the upper right corner. There is also a large user community

² “SCRA” is the Steel City Robotics Alliance. If you are not familiar with it, I highly recommend that you visit www.steelcityrobotics.org. Among other events, each year they sponsor a workshop for teams to hone their skills or learn about new topics.

³ Excel is a trademark of the Microsoft Corporation.

⁴ Microsoft Project is a trademark of the Microsoft Corporation.

on the www.projectlibre.org website that might be useful for you to join. Given the short build time, though, many questions will be time-urgent and so while the author of this document is by no means an expert in every piece of project management tools, he is willing to help where possible. You can reach him by sending an email to cyankes@msa.com. To help have your email stand out from others, please put “ProjectLibre” in the subject field.

As stated earlier, this tutorial just scratches the surface of what ProjectLibre can do. Amazon has books (paper or Kindle) available on ProjectLibre including a user manual, a tutorial, and other resources. (These books will teach you areas likely beyond what you’ll need for the robot build. As I suggested in the Workshop, given the size of the build teams and the length of the tasks you can probably skip the added complexity of the “resources” concept of assigning individuals to tasks.)

Getting Started

ProjectLibre was so simple to install on my machine that I will presume that you have done that and are ready to run it. When you first run it, you will see a license popup. While this is not a recommendation either way, I will presume that you have accepted the license and want to continue with the tutorial.

Creating a Project

Closing the “tip of the day” popup will bring you to another popup asking if you would like to open an existing project or create a new project. Select “Create Project.”

The “New Project” box is straightforward. You can name the project anything you would like and “sample project” works well for this tutorial. (Note that if you later merely “save” the project, the filename will default to your project’s name with a .pod extension. You can, of course, “save as” to give the project file a specific name.) The manager and notes fields are optional and I’ll explain the “forward scheduled” option later (but for now just leave it selected). The project’s “start date” field will default to today’s date, but to allow your sample project to follow this tutorial exactly, change the start date to July 1, 2014. As with the other date fields, you can set this value by either just typing “7/1/14” in the box or you can click the down arrow on the right side of the box which will cause a calendar to appear where you can navigate to the desired month and just click on the desired date. Whichever way you chose, set it to July 1, 2014 and click “ok”.

Normal Display

What you now see is the normal task list / Gantt chart display screen where you will spend most of your time. Since you haven’t defined any tasks yet, both the task list on the left and the Gantt chart on the right are empty. Before we add any tasks, let me warn you that it is easy to get to other screens where it isn’t obvious how to get back to the normal display. For example, in the upper left section you currently see various file options (save, open, etc.), print options and project options. Click on “Projects” in the Project section. (You will see one line for your sample project.) It is not obvious how to get back to the normal display, right? Even the back arrow to the right of the ProjectLibre logo does not get you back! Getting

back to the normal display is easy once you know how to do it. Click the “View” tab and on the far left you’ll see the Gantt icon and word. Click on that and you are back to the (currently blank) normal display for your work. Without clicking on anything other than the toolbar tabs “File”, “Task”, “Resource” and “View”, take a moment and look at the options that appear under the tabs. There is a lot there, but don’t worry as there are only a few options that you’ll need to know for the robot build.

Navigating in this Tutorial

As you looked through the tabs did you notice that the options are grouped into sections? In the previous paragraph I asked you to “click the ‘View’ tab and on the far left you’ll see the Gantt icon and word. Click on that...” That is rather wordy. From here on I’ll shorten this to tab / section / option and so moving back to the normal display is done by clicking **View | Task views | Gantt**. If the option I’m asking you to click is currently displayed (i.e. there is no need to switch to a different tab), I’ll specify the option by just saying **section | option**. Therefore, from where you are now at if I wanted you to click on Resources, I’ll ask you to click on **Resource views | Resources**. Go ahead and click on that. Now get back to our normal display.

Defining Your Project

Adding Tasks

While I urged teams during the Workshop presentation and in the slides not to have individual tasks longer than one week long, for this tutorial I’m going to ignore that suggestion for one simple reason: Having nine tasks is easier for learning a tool than is having perhaps 30 to 40 real tasks during your build. Here are the hypothetical tasks and their estimated duration (length of time to complete the task) we’ll be using:

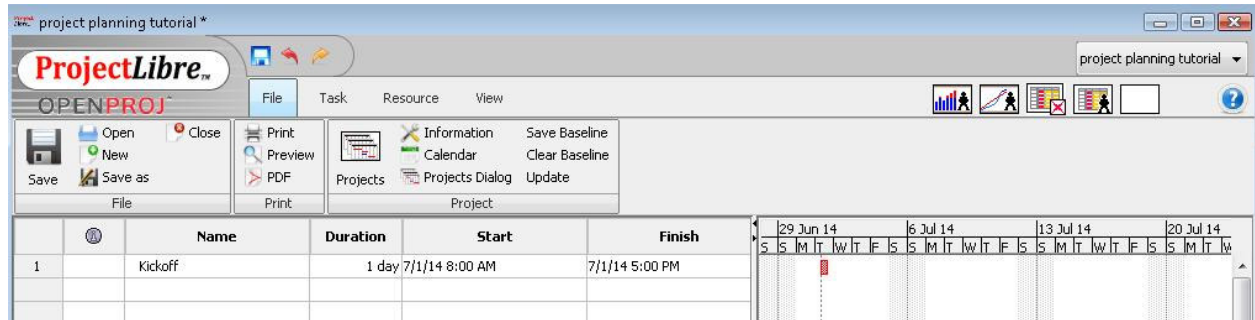
- Kickoff, 1 day duration
- Consider designs, 8 days
- Identify common design features, 1 day
- Decide final design, 1 day
- Build common features, 15 days
- Build design-specific features, 20 days
- Programming, 15 days
- Testing and rework, 10 days
- Practice, 10 days

I’m sure many of you are cringing at these tasks and the length estimates. Good! If so, you realize that these are crude and unrealistic. However, they work for the tutorial. (An actual Gantt chart used by Team 3511 in the 2013 build is included at the end of this document for an example of a real set of tasks.)

We’ll now enter the first task. In the top line under the Name column, enter “Kickoff” and press <tab>. Notice how the focus has now switched to the Duration column which was automatically filled in with “1 day?” We want the Kickoff to have a 1 day duration, but since the “?” means it is an estimate let’s make

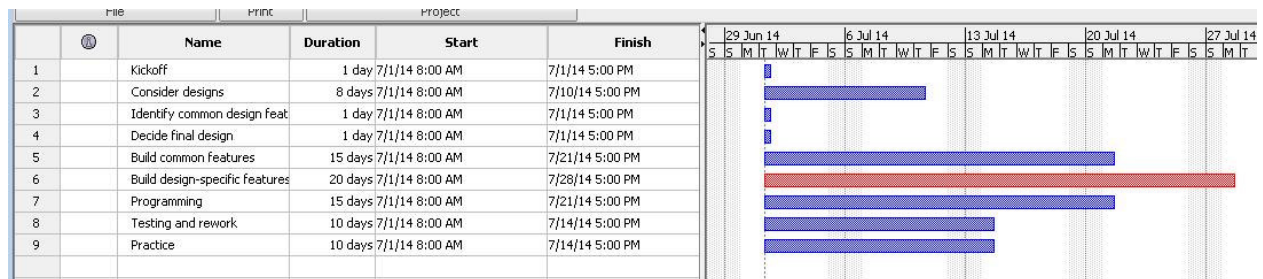
it a definite 1 day. Do this by typing “1d” into Kickoff line’s duration box. (ProjectLibre lets you enter the number and the first character of the time, such as hour, day, week, etc., if you don’t want to spell it all out.) Note that the Start and Finish boxes contain 7/1/14 8:00 AM and 7/1/14 5:00 PM respectively. ProjectLibre defaults to a traditional work week of 8am-5pm Monday-Friday. You will be shown later how to change this to better match your team’s workday schedule.

At this point, you should see the following on the screen:



If this is not what you have, verify that you correctly did the steps in the previous paragraph and make sure that when you started this project that you specified the start date as being July 1, 2014 so that our displays will match.

On the lines below the Kickoff line (which now has a “1” in the leftmost column showing this is task number 1), enter the eight remaining hypothetical task names and durations from our example. Once done with this, your screen (skipping the header section from here on) should look like this:



Notice that the “Build design-specific features” line is the only red line. As a reminder, the norm for Gantt charts (the right side of the display) is for the red line to represent the “critical path” meaning that this set of tasks, although just a single task in the example for now, represents the tasks that will ultimately determine when the project will end. Right now, since all of these tasks are scheduled to begin on the same day it is not hard to see that the longest running task (20 days long) will determine when the project will finish. It is not, of course, realistic for all of these to start on the same day so in the next section we’ll add some dependencies between these tasks.

Adding Dependencies

In project planning, a “dependency” simply means that something must happen before something else can happen. If you are going to a store alone by driving, it is intuitive that you must complete the task “walk to the car” before you can begin the task of “drive to the store.” We would then say that “drive to the

Let's consider the dependency between our first two tasks; "kickoff" and "consider designs." Since it is at the kickoff that the teams learn about that year's competition, it is reasonable to say that the team can't start considering designs until the kickoff is over. To enter this into ProjectLibre, scroll the left half of the screen to the right so that you see the "Predecessors" column. We want to say that the second task cannot start until the first task has completed, so just enter "1" under Predecessors for line 2. Observe that the first part of the display has now changed to:

		Name	Duration	Start	Finish	Pri	29 Jun 14	6 Jul 14	13 Jul
							S M T W T F S	S M T W T F S	S M
1		Kickoff	1 day	7/1/14 8:00 AM	7/1/14 5:00 PM				
2		Consider designs	8 days	7/2/14 8:00 AM	7/11/14 5:00 PM	1			

In this hypothetical example, the team is going to identify the common design features among all their design options so that they can start building those common elements early. Since we have 8 days for considering designs, let's say that these common design features will be identifiable three days before all the design considerations are done. Following the examples from the slides, the dependency for line 3 will then be "2fs-3" which means that this task can start three days before task 2 finishes. Now, could we say this dependency is "2ss+5" (meaning it can start 5 days after task 2 starts)? Of course. Deciding which way to specify the dependency is easy if you ask this question: "What should happen to this task (#3) if the other task (#2) doesn't finish on time?" If "this task" should start on time anyway, then the best way of expressing the dependency would be the "2ss+5" since even if task 2 is delayed, five days after it starts task 3 can start anyway. If, on the other hand, "this task" has to delay starting if the other task is delayed, then it would make sense to express the dependency as "2fs-3" so that if task 2 is delayed, task 3 will be delayed along with it so it still doesn't start until 3 days before task 2 has finished.

As when we entered the dependency for task 2, the Gantt chart has shifted around to reflect the new timing of these tasks. Notice one thing different in lines 1 through 3, though: The arrow comes from the start of task 2's box instead of out the end of task 1's box. This visually shows whether the dependency is

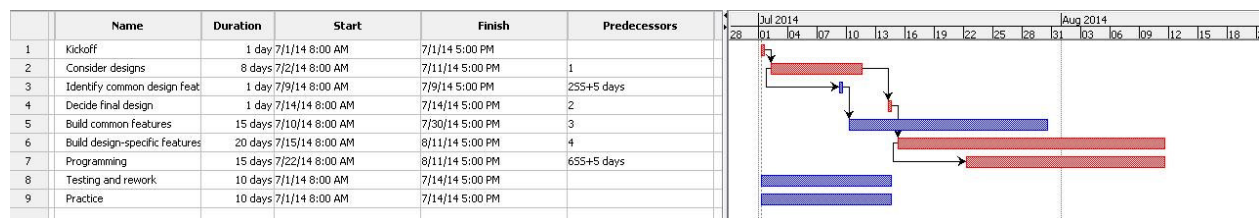
7

a finish dependency (such as from task 1 to task 2 with the arrow coming from the end of task 1's box) or whether it is a start dependency (such as from task 2 to task 3). In either case, the horizontal length of the arrow indicates how many days have to elapse before the next task can begin.

To give you some more practice at this, enter the dependencies for the following tasks:

- Decide final design: Requires “consider designs” to complete.
- Build common features: Requires “Identify common design features” to complete.
- Build design-specific features: Requires “Decide final design” to complete.
- Programming: Can be started 5 days after “Build design-specific features” has started.

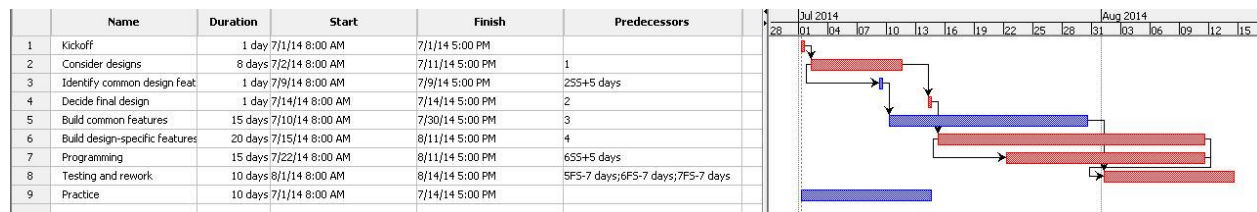
Before showing you what my screen now looks like for comparison, notice that the project has grown in length and now exceeds what (likely) conveniently fits in the Gantt chart section on the right. You can zoom in and zoom out on the schedule by pressing **Views | Zoom In** and **Views | Zoom Out** respectively. Zoom out one level and see if your screen now looks like this:



(Note that I've grabbed the line between the data and the Gantt chart sections and moved it to the right to show more data columns. You can do this by left clicking on the vertical line between these sections and sliding it right or left.)

The reason I had you stop at “Programming” was that this task and all the ones before it had only a single dependency. This is not always the case since sometimes multiple tasks have to reach certain points before a new task can start. Consider our next “Testing and rework” task. Imagine that the team has decided that it can start testing seven days before the basic robot build completes. The complication here is that given this overly high-level “testing and rework” task⁶, its start date is controlled by three other tasks; the two build tasks and the programming task. Thankfully, it is easy to describe this kind of dependency on multiple tasks: You simply list each individual dependency and separate them by semicolons. The key here is to just take them one at a time. First, the testing task can't start until seven days before the “build common features” task is done, so this is a 5fs-7. It can't start until seven days before the “build design-specific features” task is done, so this is a 6fs-7 and, finally, it can't start until seven days before the “programming” task is done which is 7fs-7. Put them together and you get the mouthful “5fs-7;6fs-7;7fs-7”. Enter this as the predecessor for task 7 and look at what happens to the Gantt chart:

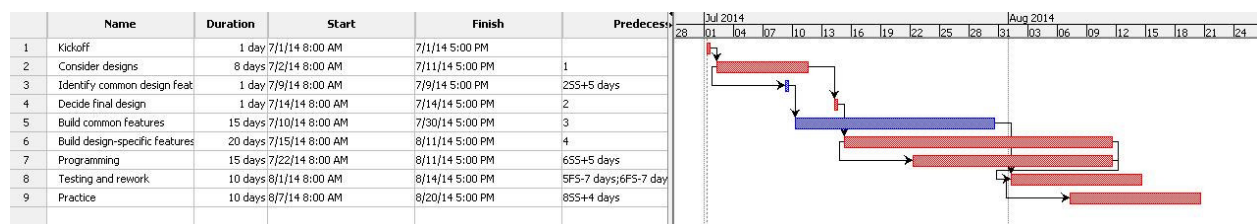
⁶ I did this on purpose for the tutorial. In reality, the “build”, “programming” and “testing / rework” lines should be at a lower level so that individual features can be tested as they are done.



See the multiple arrows that now feed into the start of the Testing box? This shows that the timing of when this task can begin is gated by multiple tasks. Now, why go through the hassle of defining three tasks as predecessors for the testing task when we can “obviously” (sic) see that just having testing start seven days before the programming ends will give us the same date? Be careful not to fall into this trap. Just because at project planning time it is “obvious” when things end at the same time, the reality of the project can delay tasks. For example, what happens if “programming” goes well and finishes early while the “build design-specific features” task takes three days longer than expected? If you just have this generic testing task gated by the completion of the programming task, the testing could end up starting too early compared to the hardware building task. While defining all the relationships for a task like this Testing task is a few minutes longer, it lets the schedule correctly adapt to any of predecessor tasks finishing either early or late. (Project planning tools will determine the last date of each of the dependencies and will use that as the start date for the task.)

Now, did you catch that I jumped right into “fs” types of relationships here? While I didn’t raise the issue in the last paragraph since it was focused on one versus multiple predecessor tasks, remember the question introduced earlier about “What should happen to this task if the other task doesn’t finish on time?” We could have defined the Testing task as beginning some number of days after each of the three build / programming tasks started, but testing is a classic case where it normally makes sense to delay starting the testing if the thing(s) being tested have a schedule slip. Therefore, testing is a common task type gated by the completion of other tasks and not the beginning.

To complete the dependencies in this example, set the Practice task to begin four days after Testing has started. The final initial schedule should look like this:



Setting Work Days

As mentioned earlier, most project planning tools default to the normal workweek. Let’s say that your team works on the robot Monday through Thursday of each week and so we want Fridays to not be counted as a workday. Adjusting the schedule of workdays is easy. Click on **Task | Calendar** and the “Change Working Calendar” box will appear. Scroll to the July 2014 month calendar and click on the “F” in the day of week line. See that all the Fridays have been highlighted (on this and all the other

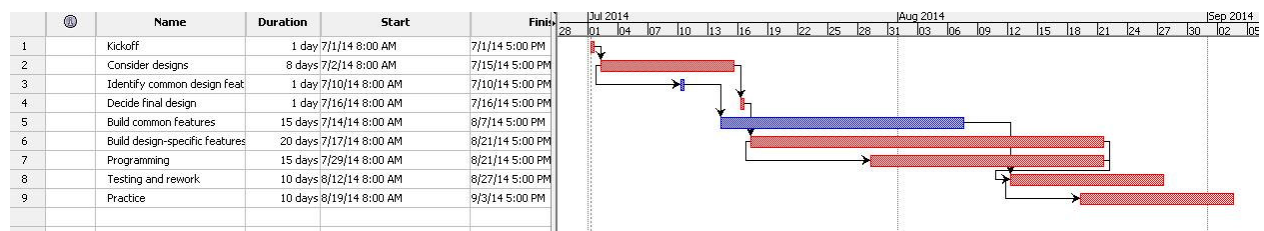
calendar pages whether shown or not). On the left side, click “Non-working time” and then “OK” at the bottom. Take a look at the ending date of the project. Now, instead of it ending at the end of the week of August 18, the project will now extend into September and if you zoom in (**Views | Zoom In**), you’ll see that Friday, Saturday and Sunday are now all grayed out to show that they aren’t work days.

Let’s now look at how to deal with specific days off. July 4th is a day that schools are closed, so will we have to adjust the schedule to show that as a non-working day? Click **Task | Calendar** again to check which day of the week July 4th 2014 is on. It is on a Friday which is already a day off for our hypothetical team, so we don’t have to tell ProjectLibre to ignore that day. That’s boring so let’s pretend that the schools are not in session on July 21, 2014 for some reason. In the “Change Working Calendar” box, click on just July 21, 2014 and click non-working time and then OK. The Gantt chart will now show that to be a non-working day and will shift everything after it to the right one (work) day.

If you want to change a non-working day to being a working day, select the day as above and click either “use default” or “non-default working time.” The difference between these is whether the day is one of your normal workdays or not. If it is a normal workday, clicking “use default” will return it back to normal. If it isn’t a normal work day (a Saturday, for example), click “non-default working time”. Either way, one of these will turn the non-workday from gray to white background and that is the one you want to use.

Using Critical Paths

After entering your real project’s tasks and their dependencies, you might find that the project’s ending date is later than when it has to complete. Pretend that the example project we’ve created has to be completed before September 1st. The Gantt chart “critical path” identification (the red lines) helps you to see which tasks to try to shorten. Here is what the project should look like in your display:



See how the “build common features” task line is blue? The reason it is blue is that the task gated by its completion (the testing task) also requires other, later completing, tasks to complete. To show this, change the duration of the “build common features” from 15 days to 12 days and look at the project’s end date. See how the end date didn’t change at all? *While shortening tasks that are not on the critical path might free up people to work on other tasks, shortening it alone will not change the project’s completion date.* (Please return “build common features” to 15 days duration now.) If you need to shorten the schedule, concentrate on the red “critical path” tasks.

While you are shortening task durations, don’t be surprised by seeing the critical / non-critical (i.e. red / blue) status of the lines change. As an example of this, change the duration of “build design-specific features” (line 6) from 20 days to 18 days. Doing this shortened a critical path task yet didn’t change the end date and, furthermore, the line turned blue. The reason this happened is that while it is no longer a critical task item (by gating when the testing task could start), another task also gates the testing task and

so the project's end date still didn't change. (Note that right now all of the tasks prior to the build task might be blue. I have seen a bug in ProjectLibre that occasionally when a red task becomes blue it accidentally, and incorrectly, sometimes paints the earlier tasks blue as well. If this happens, you can force it to recalculate all the colors by taking the last task, shorting it by a day and then putting it back to where it was.) At this point set the duration of line 6 back to 20 days and save your project file (**File | File | Save as**). Now that you can easily get back to where we currently are at in the tutorial (**File | File | Open**), feel free to play with changing durations of the various tasks to see how the critical path line moves around and how the example project's end date changes.

Once you are done experimenting with how duration changes affects the critical path, please Open the file you saved (**File | File | Open**) so that our displays are once again consistent.

Tracking the Project

What we have done thus far in the tutorial is create a project estimating how long each task will take. While it is important for planning to do this with the best estimates you can give the tasks, the estimates will rarely turn out to be perfect.⁷ Some tasks will complete faster than anticipated while others will take longer. Dealing with this changing reality is one of the main benefits of using project planning software. In this section, we'll explore how to update the project's progress.

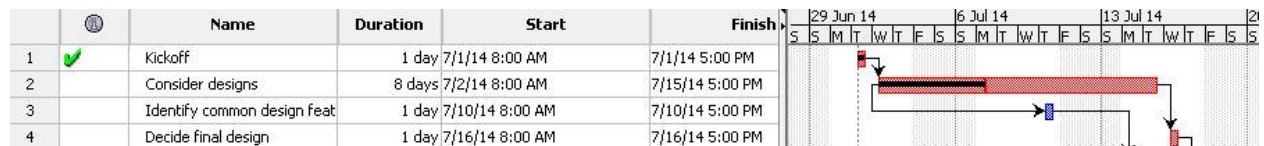
Task Completion Percentage

While the project is ongoing, you will gain much more insight into the project schedule if you periodically update ProjectLibre with the status of the current tasks. You can assign a completion percentage of 0% to 100% to any task and the program will then use this information to help give you a better estimate of the remaining work and the anticipated project completion date.

Let's pretend in our example that the kickoff has completed. The default column to the left of the "Name" column shows progress and if you double-click on a position, the "Task Information" box pops up for that task. Go ahead and bring up this box for the Kickoff task.

There are many things that can be set for the task as you'll see if you click through the various tabs. (Nearly all of which I suggest ignoring for your first use of the tool on a project as straightforward from a project management perspective as building the robot. I'll briefly explain "resources" later on.) Returning back to the "General" tab, enter 100 in the "Percent Complete" box and press "close." Note the green checkmark on the Kickoff line: This shows that the task on that line is complete. Now, while it is hard to see since the Kickoff task is only one day long, if you zoom in a bit on the Gantt chart you can see a black horizontal line through that task's red box. As you'll see in this next step, the black line represents the percent completion and doesn't only appear when the task is done. To see this, set the "consider designs" task to be 30% done. The black line for that task goes into Monday, July 7th as seen here:

⁷ Helmuth von Moltke the Elder, the Prussian Army Chief of Staff for many years in the late 1800s, is credited with saying "no plan survives first contact with the enemy." While he was describing military war plans, project planning has the same attribute that reality rarely, if ever, goes exactly as planned so don't fear changes.



If we're not yet at July 7th, congratulations, this task is ahead of schedule but what happens if "today" is Wednesday July 9th where this task now appears to be over a day behind schedule? This is the topic of the next section. Before we move into the next topic, however, I will reinforce and explain here the message on the slides that smaller tasks are better than larger tasks. The benefit of smaller tasks is that it is easier to see if they are fully done or not. Let's use an example of opposite ends of the spectrum where one project has 10 three day tasks and another project has a single 30 day task doing the same total work. After nine days it is easy for the first project's team to know if the first three tasks are done or not since they are separately defined tasks. After nine days, how easy is it for the second team to estimate their percent completion of the one large task? In general, the larger the task the harder it is to easily and accurately track its progress and errors can be major. Smaller tasks naturally lend themselves to a more definite understanding of what is, and what isn't, yet done and errors are less significant. (Estimating the completion percentage wrong by 10% on a 30 day task is a large schedule swing when it is discovered but misestimating a 3 day task by 10% is hardly noticeable.) Do yourself a favor and try to keep your tasks small.

Adjusting Late Tasks

Adjusting the schedule to account for tasks that are running behind can be one of the hardest things to do because it requires the team to acknowledge that some part of the project is behind schedule. At the same time is one of the most important things the team's "keeper of the schedule" can do. Repeating myself since this is on the slide set's annotations, the importance of adjusting the schedule for late tasks is hard to understate since doing this is how the team can recognize what I call the "current reality path" and can make changes if that path does not point to success. The earlier you can discover that a task is running late the more time you have in which to correct it, so avoid the temptation to give the team another day or two to make it up before adjusting the schedule. Besides, the team already knows that the task is behind schedule, so not adjusting the schedule doesn't help anyone. (And if the team does make up the work, the team can celebrate and the schedule can be readjusted as will be demonstrated below.)

Sorry for what might appear to have been a soapbox, but hopefully this conveys the importance of facing schedule reality. Ok, back to the mechanics of adjusting the schedule. ☺

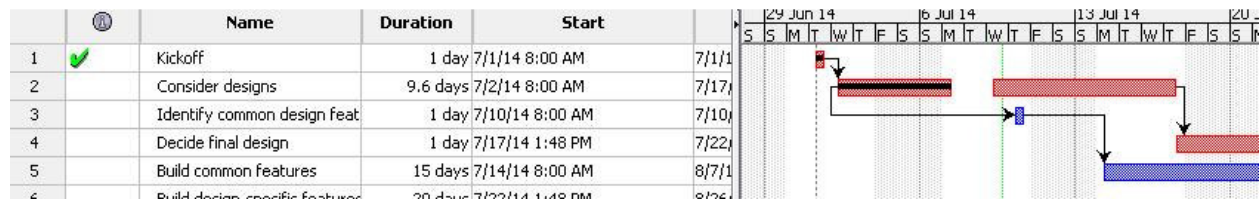
Going back to the tutorial example, "consider designs" was marked as being 30% done and is a bit over a day behind schedule since today is supposedly July 9th. What project planning tools give you is an easy way of saying that any work not yet done should start on a certain day. This lets us take the remaining 70% in this task and shift it to the right (timewise) to begin today. Since this is a project-wide setting, it isn't under the "task" tab in ProjectLibre, but rather is at **File | Project | Update**. Click on this. The "Update Project" box appears. WARNING, this box has what I believe to be an unusual default, namely adjusting every task to whatever percent completion they should be at the start of tomorrow morning's work day. For projects with a small number of tasks being active simultaneously, such as the robot build, I suggest talking to the team to find out each tasks' real percentage completion instead of using this default setting.

To adjust the schedule for tasks that are late, select the “Reschedule remaining work after” button and set the appropriate date.

Now, a note about the “appropriate date”: The date is likely one different from what you might expect. Read the wording of that button we selected carefully and note the “work after” part. If you put in today’s date, it will reschedule the remaining work to begin tomorrow which makes sense if your team reviews the schedule at the end of the day’s build activity. However, if your team reviews the schedule at the start of the day’s activities, you’ll have to put in *yesterday’s* date to have the remaining work start today on the schedule. You’ll get the hang of it quickly.

One more warning. Save your project file (**File | File | Save**) before adjusting the remaining work to start on a certain day. Sometimes, most notably if you are trying to undo a gap created by a bad date, I’ve seen the program apparently not display the proper task relationships. Save your work before you do the “update” task in the next paragraph. (And, of course, as with any program saving your work often is good. It is just good to do it before this function in particular.)

Going back to the tutorial example, we will presume the team reviews the schedule at the start of the day and that today is the 9th, so set the date to July 8, 2014 and press “OK”. You’ll now see this gap in the Gantt chart for task 2:



Additionally, you’ll see that the project has now slipped out by a few days since task 2 was on the critical path, thus making it longer makes the entire project longer.

Oddly, the adjustment feature does not also work the other way.⁸ Imagine that the team worked hard on this “consider designs” task on Wednesday the 9th and completed it. Set its completion status to 100% and then update the project for all remaining work to begin after July 9, 2014. Other than task 2 having a full horizontal black bar in it and the green checkmark appearing, nothing else happened. The easiest way of getting task 4 (“decide final design”) to show as being start-able on July 10th is to remove task 4’s predecessor dependency on task 2 and hit <tab>. This will cause task 4 to be moved back to the beginning of the entire project and then do the update outlined above to have all remaining work starting after today (i.e. July 9, 2014). That makes task 4’s start date be July 10, 2014 and adjusts the rest of the schedule accordingly.

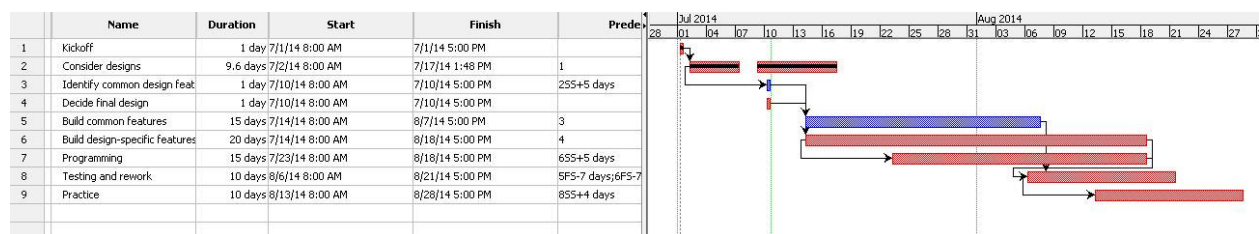
The downside of breaking these predecessors to account for tasks completed early is that you won’t have them on the schedule to act as next year’s template. If you double-click in the status column for the task (the column that has the green checkmarks for tasks 1 and 2 at this point), the “Task Information” box has a “Notes” tab at the far right. If you select that can you can make a note to yourself that this task used to

⁸ I suspect its definition of “remaining work” means what should have been done, but hasn’t yet been done, from the start of the project to today. Future items aren’t yet late and thus must not count as “remaining.”

have another task as a predecessor. This would make it easier to reuse this project as a template next year.

Wrap-up

At this point, your screen should look like this (zooming out a bit):



If it does, congratulations! You probably now know all you need to know about project planning tools to apply it to a project of the size of the robot build. If your screen doesn't look like this, go back and review the steps with a particular eye to the calendar dates being entered. If you are still having problems (or just feel that I explained something in a confusing way), feel free to send me a question at cyankes@msa.com and, to help me more easily see your question amongst my emails, please put "ProjectLibre" somewhere in the subject field.

Deferred Topics

Creating a Project with Forward versus Backwards Scheduled Tasks

Back at the beginning of this tutorial when you were creating the project, I asked you to leave the "Forward scheduled" box selected and to enter July 1, 2014 in the Start Date field. I mentioned that I'd explain what this "forward scheduling" means later on. The reason I did this was it might have been confusing to describe the "forward" and "backward" options before you had some experience working with tasks.

Throughout this tutorial, we've used what is called "forward scheduling" which is another way of saying "if the project starts on this date and has these tasks, when will the project be done?" When we added tasks or adjusted durations, therefore, we weren't changing when the project was starting but rather we were changing when the project would end. "Backward scheduling" is the opposite and essentially asks the question "when do we have to start this project with these tasks if it has to be done on a certain day?" In backwards scheduling, if you add tasks to the critical path or increase their duration, the mandatory project finish date remains fixed but the start date will move earlier on the calendar.

Both of these styles have environments in which they make sense. “Forward scheduling” is quite common on projects with definite start dates but indefinite end dates and is often used to give upper management an idea of when the project might finish (and thus an estimate of its costs). “Backwards scheduling” is often used in environments where there are contractual completion dates and the company performing the work wants to know when the various parts of the project have to start in order to meet the contract.

Oddly, the robotics build is both.

You have both a definite start date, kickoff, and a definite “stop build” date. In this sense, either forward or backwards scheduling can work for the robotics project. In either case, the quality of your schedule depends upon the team’s ability to identify and estimate the duration of the tasks and in either case you have to adjust for tasks being ahead or behind schedule. Therefore, you can choose either method. One important difference between these approaches, though, is the impact of a task taking longer than expected. In backwards scheduling, tasks are delayed as long as possible and are generally started at the last possible moment. This works well if you have a solid understanding of how long something will take. (For example, while constructing a building a general contractor with experience will likely know down to the day how long it will take to pour the concrete foundation.) If you do choose to use backwards scheduling, I suggest that you start to work on tasks as early as you can anyway to allow for later schedule slips. Personally, I’d use forward scheduling for the robot build.

New Topics in the Second Version of the Tutorial

Context for Version One

This tutorial was originally written for a specific audience: Teams involved in the *FIRST* Robotics Competition (FRC) program. As such, the context for the tutorial was not described. As this tutorial is now finding use in other environments, I'll take a moment to explain the context for the prior discussion and examples. *FIRST* (“For Inspiration and Recognition in Science and Technology”) is a US-based, but now international, organization whose goal is to inspire students to explore the science and technology fields by hosting four levels of robotics competitions. (See www.usfirst.org for more information.) At the FRC level, the student teams have only six weeks from the date that the year’s challenge is announced to design and substantially complete a large, up to 120 pound (54.5 kg) 5 foot (1.5 m) tall robot that competes against other teams’ robots. As an understatement, this six-week “build period” is an incredibly busy time for the teams and so as part of helping the local teams gather the skills they will need, this tutorial was written to help teach them the basics of project management. This is why there are so many references to building robots.

FIRST sponsors four levels of robotics starting with Junior *FIRST* Lego League (JrFLL) for children as young as five or six years old on up to the FRC level for those in the last three years before moving on to college. Any reader wishing more information about these programs can contact the author.

Resources

Dealing with “resources” was not included in the original version of this tutorial as most of the robot builds involve teams that are small enough to not need this additional level of project control. However, a discussion of managing resources was requested and thus is added to this version.

The first version of this tutorial focused on managing tasks: how to define them, assign dependencies from one to another and how to see whether the project is on track or not and, if it is not ending soon enough, how to see which tasks are the “critical path” tasks that determine when the project will end. Managing a project by keeping track of just the tasks is sufficient when dealing with a small project like the robot build period where it is just six weeks long, has a reasonably small number of tasks and a small-to-moderate number of team members with each generally working in a specific area such as programming, game strategy, pneumatics, etc. Managing a complex project (where the complexity comes either from the number of tasks, the number of people involved or having people working on multiple tasks) is more difficult if the focus is only on the tasks. As an example of the potential problems, imagine managing a project with three tasks that can be active at the same time, Task_1, Task_2, and Task_3, and while the other team members are best focused on certain task areas, the team has a person that is equally qualified to help with any of these three tasks. Which task should this person be assigned to? If the project management’s focus is only on the tasks, it could be difficult to make the correct decision and, worse yet, in a sufficiently complicated environment this person might accidentally be assigned to work half of their time on each of these three tasks! (This does not mean that the project manager knowingly did this. Imagine a situation where Task_1 and Task_2 are both active tasks and it is believed that Task_3 will not start for quite a while due to predecessor tasks. It could be reasonable to assign this person to

work half their time in all three of these tasks since only two are currently active. What happens if Task_3's predecessor completes earlier than anticipated? All of a sudden, this person has three active tasks to work on 50% each.) Managing the project only by tasks risks that the group members will not be efficiently used or risks having the schedule based on an impossible level of effort required from group members. Resource management is the part of project management that attempts to avoid either of these situations.

So what are “resources”? Recall that tasks are the individual pieces of the project. “Resources” are what is needed to perform and complete the task and normally include people but can also include special equipment needed for the task. (For example, a high quality 6D computer-controlled lathe that can be rented at an acceptably low cost to the project, but is available for only one month at a particular point during the project's schedule.) “Managing resources”, therefore, is simply ensuring that the required people and equipment are available to work on the task. As you can imagine, if the resources are not available for when the task is scheduled there are two basic ways of solving the problem:

1. Move the task to when the resources are available. Using the computer-controlled lathe example, defining the task to be dependent on the lathe and the lathe being defined as being available in that one month will allow it to be used when it can be obtained inexpensively. However, this might not be practical if the task is either performed so late that it is delaying the rest of the project or if the one month of lathe availability is before the project is ready to use it.
2. Move the resources to when the task has to be performed. If the task has to be done at a certain time, the alternative is to make the resources available at that time. This could be reassigning people off of other tasks so that they can work on this one or, continuing the lathe example, might entail the project team deciding that even though the lathe is inexpensive in that one month, it might be better to pay more to rent a different lathe at a time more convenient to the project.

While managing resources is required for large projects, as you can imagine it can also potentially be time-consuming. However, managing both resources and tasks gives a more complete picture of the project and, accordingly, a higher degree of confidence in the schedule, the effort required to complete the project and the knowledge that special equipment will be available when needed. That last sentence sums up the goals of project management. The rest of this section will show examples of using *ProjectLibre* for both task and resource management.

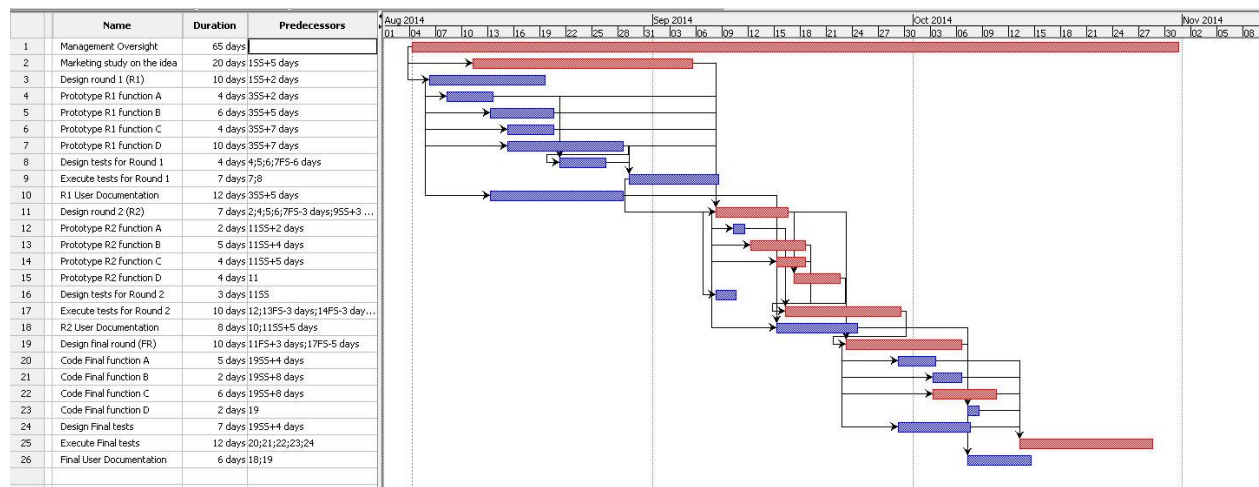
To provide a wider set of tasks in which to explore resource management, the robot build example used previously will not be used in this section. Rather, we'll use an example of writing and marketing a mobile smartphone app. The project will begin on August 4, 2014 (the first Monday in August) and the team is made up of the following people:

- Team Member (“TM”) 1: Manager for the team and can help with programming, but programs at only 50% the rate of the other programmers and cannot do function D. Available 30 hours per week.
- TM2: Good marketing person and can offer ideas for what the app should do. Available 20 hours per week.
- TM3: Good programmer, but only understands areas A, B, and C. Available 30 hours a week.
- TM4: Software architect and can program any app area. Available 40 hours per week.

- TM5: Good at designing tests. Available 10 hours per week.
- TM6: Good at executing tests. Available 40 hours per week.
- TM7: User Documentation Writer. Available 20 hours per week.

As with almost any team, the people in the team have areas they are good at and others that they shouldn't be assigned to.

This mobile smartphone app will go through three rapid design / code / test cycles. Skipping many of the tasks that would be actually be required in order to simplify this example (most notably fixing the bugs found in testing), here is the initial task-oriented plan for this project:



This simplified project should, in an ideal world, complete towards the end of October. An ideal world, however, is defined as one in which we have infinite resources and so every task can begin as soon as its predecessor requirements are met and will complete in exactly the number of days specified in the duration field. Sad to say, the ideal world doesn't exist: We have seven team members, and only a few of those can work on this project full-time. Clicking over to the "Resource" command and clicking on the "Resources" icon on the left side of the options line, we will define the members of the team in this way:

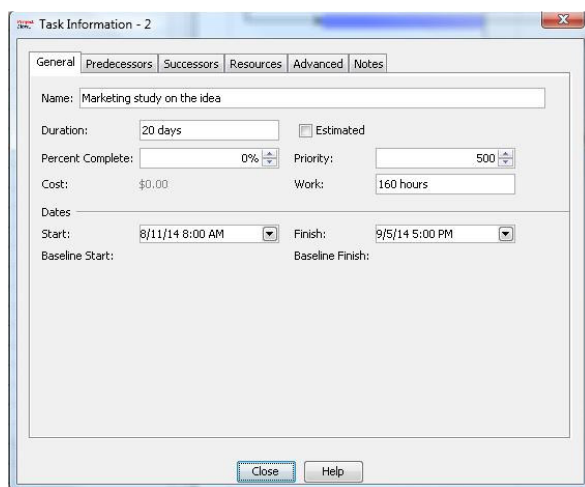
ProjectLibre™										
OPENPROJ™										
Resources		Views		Clipboard		Resource				
Name	RBS	Type	E-mail Ad...	Material ...	Initials	Group	Max. Units	Standard Rate	Base Calendar	
1 TM1	Management	Work			TM1	Manager	75%	\$60.00/hour	Standard	
2 TM2	Marketing	Work			TM2	Marketing	50%	\$40.00/hour	Standard	
3 TM3	Programming	Work			TM3	Programming	75%	\$35.00/hour	Standard	
4 TM4	Programming	Work			TM4	Design, Programming	100%	\$60.00/hour	Standard	
5 TM5	Test Design	Work			TM5	Testing	25%	\$25.00/hour	Standard	
6 TM6	Testing	Work			TM6	Testing	100%	\$18.00/hour	Standard	
7 TM7	Writer	Work			TM7	User Documentation	50%	\$40.00/hour	Standard	

Since we cannot use all these team members at a full-time rate (based on the common United States full-time rate of 40 hours per week), this is a good time to describe the relationship between *Duration*, *Work*,

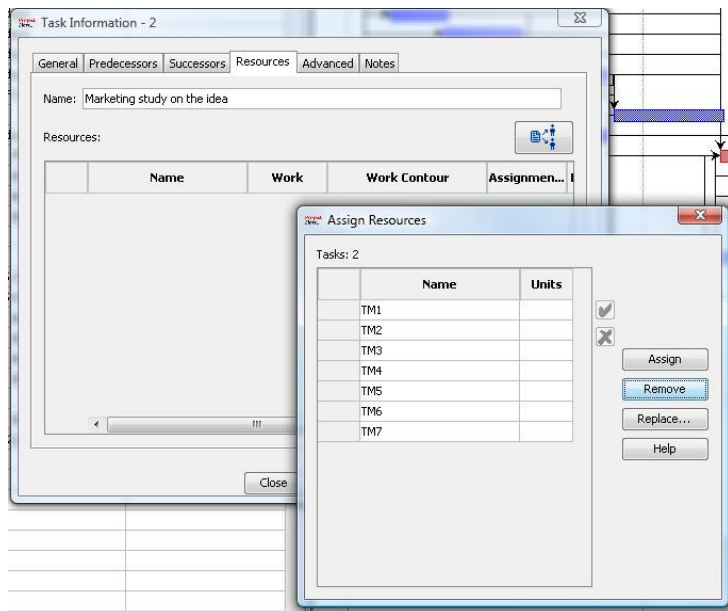
and *Units*. *Work* is the amount of effort required to complete the task. For example, someone might look at a specification for a software routine and estimate that it will take 20 hours of effort, or work, to complete the task. In terms of calendar time, how long will it take to complete this task? It could be completed in one day if the programmer is willing to not sleep. It could take 20 days to complete if the programmer worked on it one hour per day. (Ignoring, for this example, that these are not really equivalent given how long it takes to remember where the person stopped working the day before.) The difference between these is the *Units* being applied to the task. In a 40 hour per week environment, working on it full-time would be a *Unit* of 100%. Working on it 20 hours out of each 40-hour week would be 50%. (While *Unit* is the recognized industry term for this, you can think of this as a percentage of their week's work going towards this task. You might think that "Rate" would be a good word, but it is already used to mean the amount of money the workers get per quantity of time as you can see in the previous screen-shot.)

The *Duration*, *Work*, and *Units* interact in the following manner: $Duration * Units = Work$. This equation shows the result of changing any one of these. If you double the effort (*Units*) put onto a task, it is reasonable to expect it to be completed in half the time (*Duration*). In the same way, doubling the amount of work to be completed (*Work*) without changing the available time (*Duration*) means that the effort (*Units*) has to be doubled or else the schedule will not be met. Remembering that $Duration * Units = Work$ is one of the keys to resource management.

The first task we'll assign to a person is Task 2, the marketing study. (Task 1 is being skipped on purpose for now and will be discussed later.) This task is expected to take 20 work days if the person it is assigned to is working on it full-time. In the Gantt chart view, double-click on the box for Task 2 in the "Resource Names" column. The following box appears:

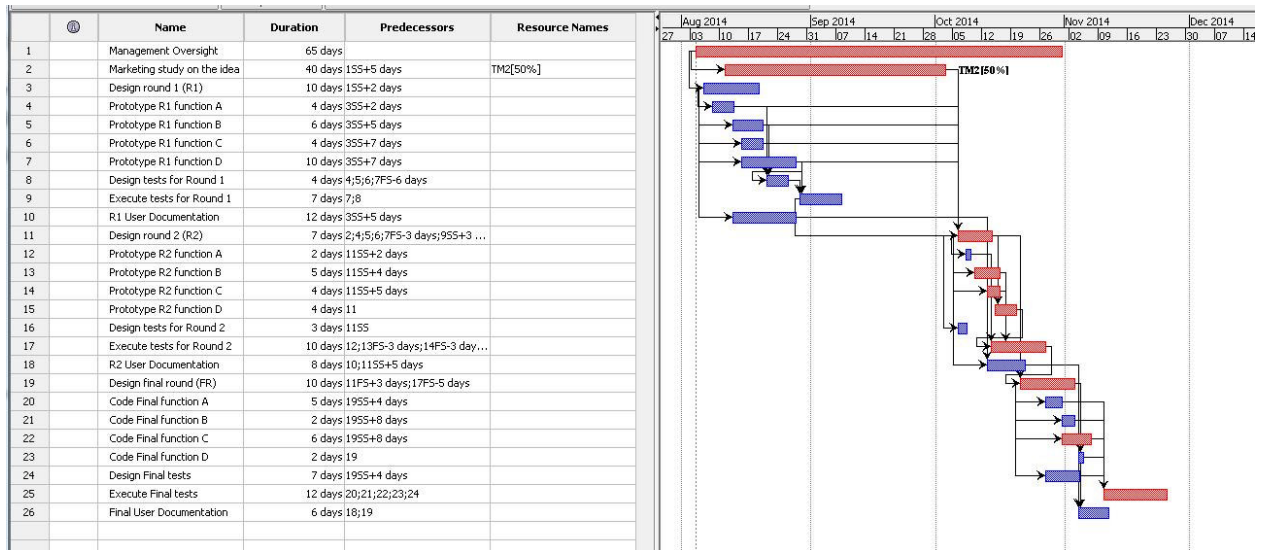


This should not be a surprise since this task was initially defined as taking 20 days to complete and so this box shows the *Work* as being 160 hours, or 20 days * 8 hour days. Clicking on the "Resources" tab gives us an essentially blank box showing that nobody has been assigned to this task yet. Clicking on the box whose icon looks like a piece of paper with arrows to two people (towards the upper right) gives this:



Recall that the team members were given the boring names TM1 through TM7. This box lists all of the team members so that you can assign who will do this marketing project. Since TM2 is this team's marketing person, click on TM2's box in the Units column and you will see it change to 100%. Press "Assign" and you will see the TM2 line turn green to show the assignment. If you click the "x" to leave this box and click "Close" on the "Task Information" box behind it, you will see that TM2's name is now to the right of the task on the Gantt chart. Other than that, nothing changed, but should it have changed? Remember the important $Duration * Units = Work$ equation. When the tasks were first defined ProjectLibre presumed that each task would be worked on full-time ($Units = 100\%$) and since TM2 was just assigned to this task 100%, neither the *Duration* or *Work* had to be changed and so the project's schedule remained the same. However, the schedule is now unrealistic since this task is expecting TM2 to work on it full-time yet we know from the team member listing (above) that TM2 only works 20 hours out of each normally 40-hour week, or at a *Units* rate of 50%.

We can correct this by first double-clicking the Task 2 box in the "Resource Names" column and clicking on the blue paper-to-people icon again. TM2's *Units* value is 100%. If you click on the "Units" column to the right of TM2, it will likely change to 150%. We want this to be 50% since TM2 only works at a half-time rate, so you can either achieve this by clicking the down-arrow several times or else by putting the cursor in the 150% box, delete what is there and enter 50. (The software expects this to be a percentage so you don't have to type in the "%".) You can either click on "Assign" or press "return". Press "x" to close this box and click "Close" to return back to the Gantt chart. Note that TM2's name has been changed in both the "Resource Names" column and to the right of the task's bar to TM2[50%] so that you can easily see the *Units* that TM2 is expected to put onto this task. Zooming out to get the whole chart on the screen, here is what the Gantt chart now looks like:



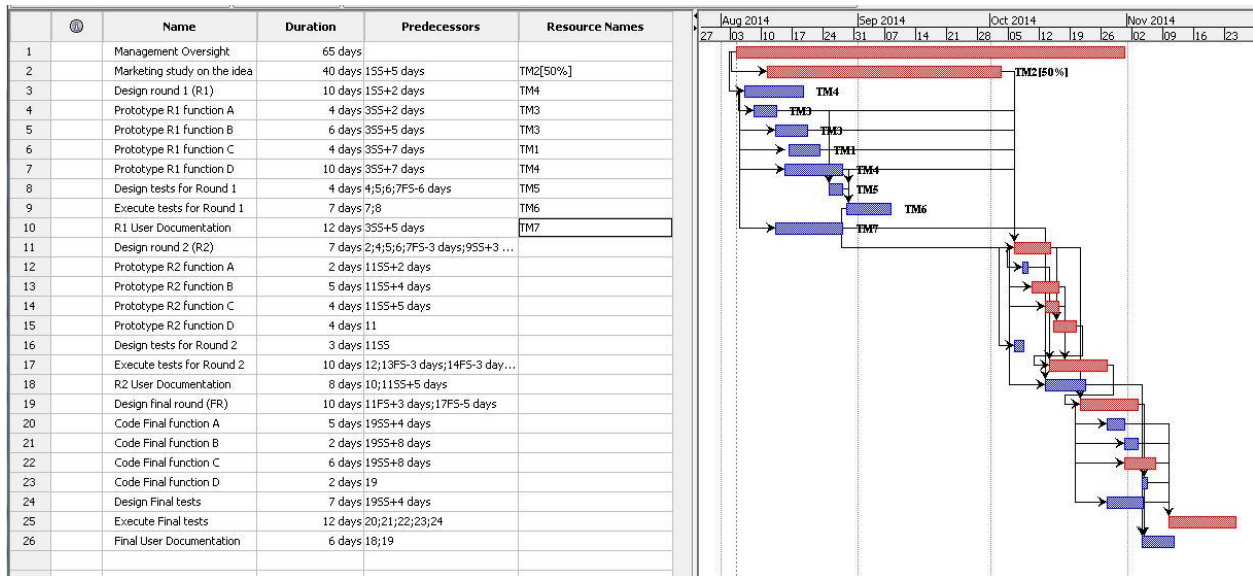
The project will now complete towards the end of November instead of the end of October! The reason for this is simple: The tasks are defined so that the second round of design is not started until the marketing study is completed. We have told the software that Task 2 is being done by someone who only works 20 hours per week and so it recalculated the ending date (remember: $Duration * Units = Work$) and adjusted the schedule of the other tasks that depend upon it. Is this new Task 2 completion time a problem? We won't know until we assign the resources to the rest of the tasks.

To simplify the tutorial as this software development effort goes through three similar design, code and test cycles, we will assign team members only to Tasks 1 through 10. As an exercise, you can recreate the project plan where this tutorial ends, continue assigning people to the second and third development cycles, and see the schedule effects of bringing other people onto the team.

We will next assign people to Tasks 3 through 10. (Task 1 is still being held off for discussion later on in this tutorial.) Recall that we have one software architect (TM4), TM3 can program but only functions A, B, and C, and TM1 can program but at a slow rate. The decision, therefore, of which team member will do the design task (Task 3) is simple, TM4, and we will make that allocation. We have four programming tasks, 4 through 7, one involving function D (Task 7) that only TM4 can code so we will allocate TM4 to that task. Since TM1 is slow, we will assign Task 6 to that person and the other two to TM3. (Making all assignments at the 100% *Units* level for now.)

Moving beyond the programmers, TM5 is our test designer and so is assigned Task 8, TM6 executes tests and so is assigned Task 9 and, finally, TM7 is assigned the user documentation writing in Task 10.

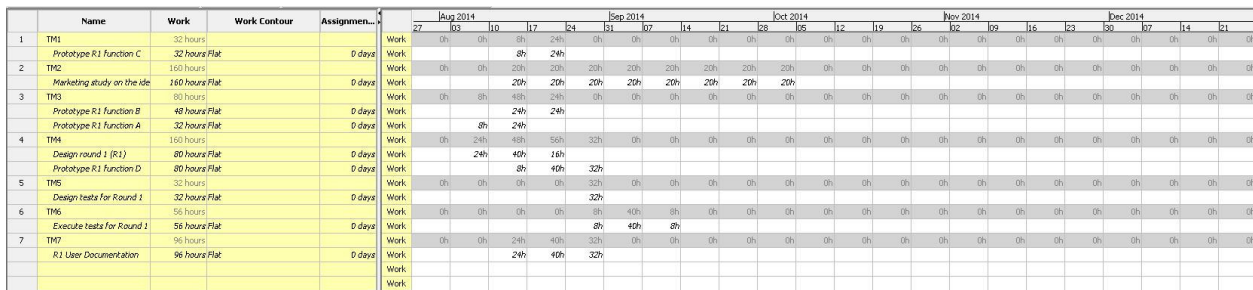
Except for now having the names beside the task's bars, the Gantt chart remains unchanged:



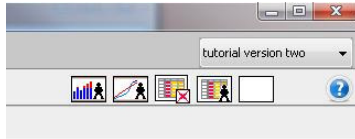
While the schedule hasn't changed, there are problems in the project. Look at TM4 as an example. This person cannot be working full-time on designing while also working full time on function D. Similarly, nobody on the project except for TM4 and TM6 are full-time. Project management software gives you easy ways of finding these kinds of problems. If you click on "Resource" on the top row of options (where it says "File Task Resource View") you will see the following options:



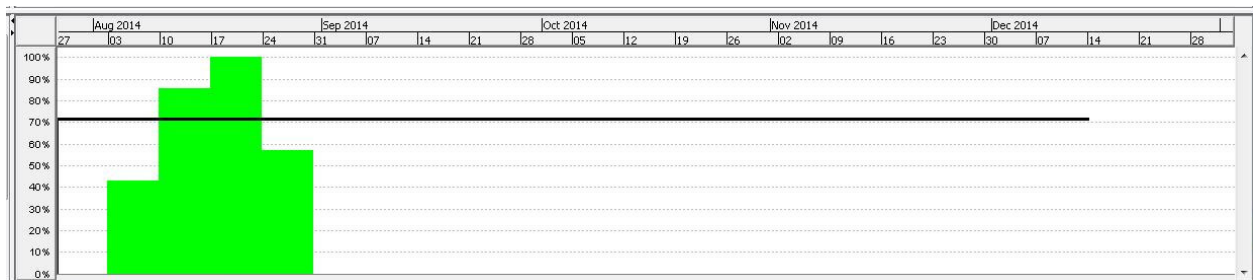
If you click on "Resource Usage" (in the "Views" box on the left) you will see a day-to-day listing of how many hours each person is expected to work in each week. For each person, the gray-shaded line is their total for the week, due to my zoom level, and the non-shaded lines under it shows the different tasks that person is to work on during that week. While it has all the information, a chart like this is not intuitive to use if the goal is to see if anyone is over- or under-used on the project:



This software, as does nearly all project management software, can graph the commitments that have been put onto team members. In the upper right corner of the screen you'll see these buttons:



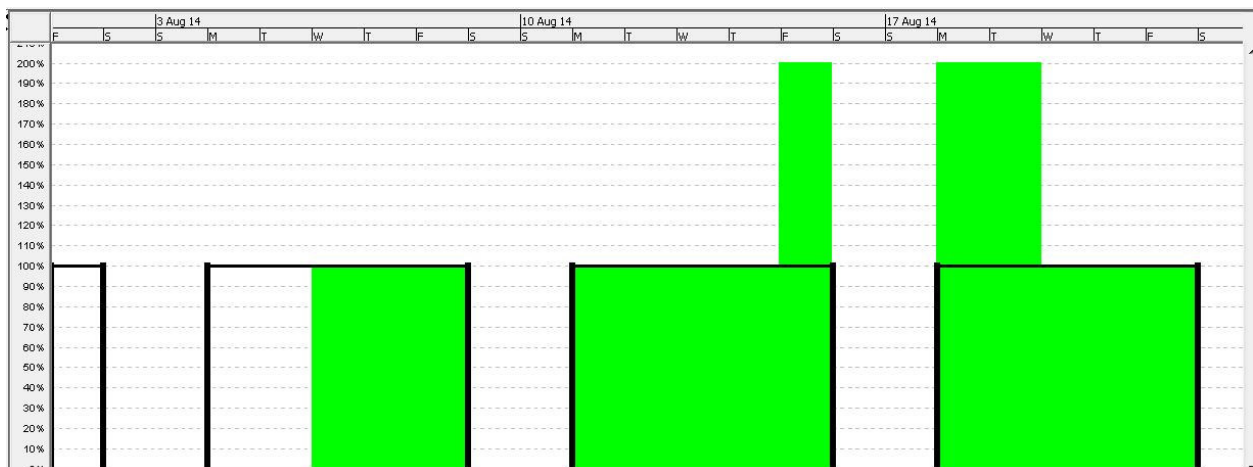
Click on the left most of these buttons and the histogram boxes will appear at the bottom of the screen. If you click on a team member's number to the left of their name on the gray-shaded line, you will see a graph appear showing their work levels and availability similar to this for TM4:



As the legend to the left shows, green represents the amount of time this person is spending on this project while the black line shows the person's availability over the course of the week (in this case). Why is the availability line only a bit over 70% when TM4 works on the project full-time? "Full-time" means 40 hours a week normally on Monday through Friday, but a week includes Saturday and Sunday. The software is averaging all seven days of the week and shows, in this case, that TM4 is available for 70% of the entire week, including weekends.

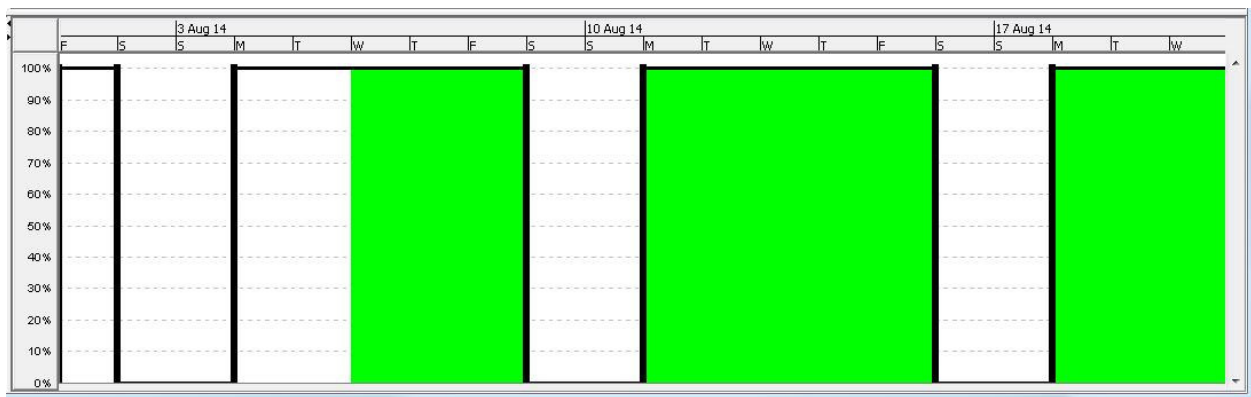
(As an aside that this tutorial will not really get into, but *ProjectLibre* allows you to define custom schedules for each team member so that they are not all expected to work the standard 40-hour Monday through Friday 8 hours per day schedule. If you want to explore this, click on Resource | Views | Resources and if you double-click on a team member's name you will see a "Resource Information" box. Clicking the calendar icon to the right of the "Base Calendar" selector allows you to customize work schedules for this person, including adding in when the person will be out for vacation time.)

Returning to the resource histogram, if you click "zoom-in" enough times to show the information at a daily level, you will see this for TM4 at the start of the project:



This level of detail shows that TM4 is available Monday through Friday at a 100% *Unit* rate (the black line), but look at what happens between August 15th and August 19th: TM4 is expected to do two people's worth of work (200% *Unit* rate) for those three working days. This shows a scheduling problem. Even though the tasks allow for the prototyping of round 1's function D to start before the design task is completed, we have a practical limitation that the design has to be completed first so that TM4 is not overloaded. (We can assign TM4 on a 50%/50% basis between these two tasks during the overlap period, but this is a more advanced level of usage. I suggest getting comfortable with the basic features first before attempting overlaps like this.)

There are two ways of resolving this situation. The first is to change Task 7's predecessor dependency in the Gantt chart screen that it cannot start until Task 3 has completed. (It is currently "3SS+7" meaning it can start 7 days after task 3 begins.) This is not the preferred method. A different approach is to use the "Leveling Delay" option for the task. Double-clicking on the task will give you the "Resource Information" box. Scroll over to the right and you will see the column entitled "Leveling Delay." As the name suggests, this is a delay that can introduced before a task is to begin to help with leveling the workload for a team member. Changing it to "3 days" for Task 7 delays it by 3 days and gives us this resource utilization for TM4:



TM4 will be much happier with this allocation. Now, why did I suggest using the "Leveling Delay" feature instead of changing the task's predecessor? If you go back to the resource chart you will now see this for TM4:

4	TM4	160 hours				Work	0h	0h
	Design round 1 (R1)	80 hours:Flat		0 days	0 days:Rate A	Work		
	Prototype R1 Function D	80 hours:Flat		0 days	3 days:Rate A	Work		

While the column headers were not included in this, the "3 days" in the Leveling Delay column stands out as an obvious number that is hard to forget while changing Task 7's predecessor rule can easily be forgotten. The difference comes in if another team member is added that can do Task 7. If that task's predecessor rule is changed the new team member won't start the task until the design task is complete as that is what the predecessor rule now states. That is not what we want! Changing the predecessor rules to level the workload puts two very different concepts together: The true dependencies between the tasks and the convenience of delaying a task. To avoid many potential problems if a new team member can be added, keep the predecessor rules defining the true relationships between the tasks and use the leveling delay to insert possibly temporary delays before certain tasks.

If you return back to the Resource Usage chart, click on TM7 to show another issue. If you zoom in to the daily level and look at the time between August 13th and August 28th, this person shows 100% usage but only has a 50% availability. This person, like several others on this project team, work less than full-time and TM7 only works at a 50% rate. If you go back to the Gantt chart and double-click on task 10, click again on the blue paper-to-people icon and in the “Assign Resources” box change TM7’s *Units* to 50%. The Gantt chart now shows that the R1 User Documentation task (Task 10) will complete on September 15th and the workload histogram at the bottom shows an appropriate workload for TM7. (Incidentally, if you are wondering by now how to get rid of the graphs at the bottom of the screen, clicking on the histogram button in the upper right again – the button you pressed to bring this up on the screen – hides the graphs.)

At this point, with the exception of Task 1 identifying whether any other team members are overloaded during this first round of the application development and leveling the workload for those team members will be left as an exercise for the reader.

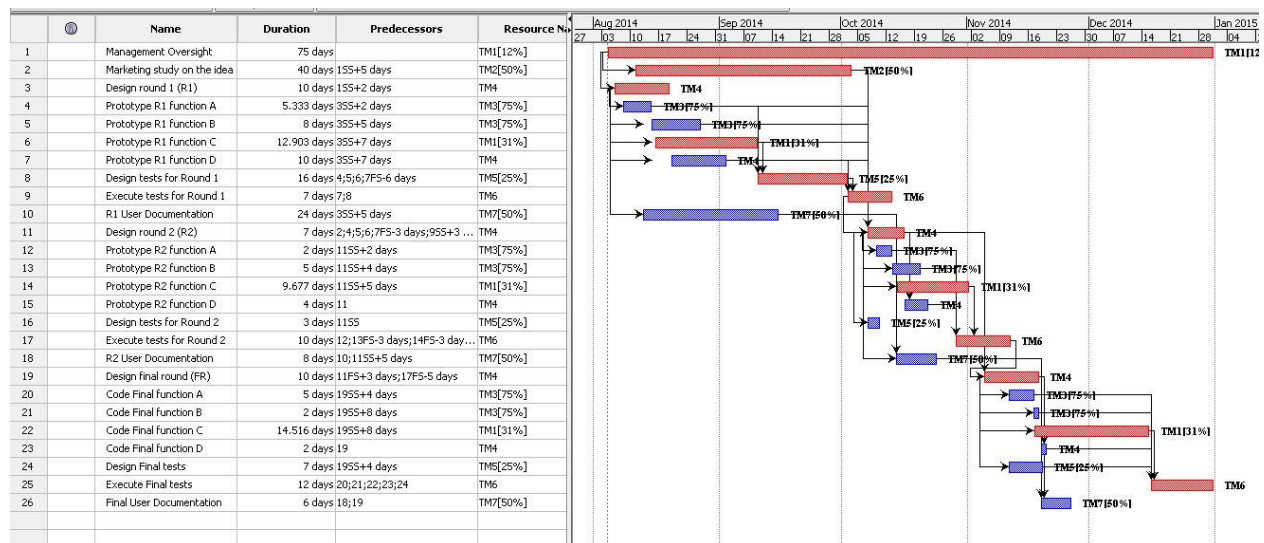
Let us now, finally, turn our attention to Task 1. All of the other tasks have a defined amount of required work and so the question becomes when and who can best perform that task. Management oversight, on the other hand, goes for the entire duration of the project. This task initially had 65 days of duration (since that was how long the project was going to run before factoring in resources), but with what you now know you can see that this is incorrect. When we entered tasks, *ProjectLibre* presumed that the *Duration* was the same as the *Work*. Even if the project did take 65 calendar days, a small project like this would likely not need 65 full-time days of management oversight. Therefore, how do we define a task that extends for a period of time? There are a few ways of doing this. The first is to change the task from the default “Fixed Work” type of task to be a “Fixed Duration” task before assigning anyone to the task. To do this, double-click on the task and in the “Task Information” box, click on the “Advanced” tab. In the “Type” pull-down, select “Fixed Duration” and make sure that the “Effort Driven” box is not selected. Now go back to the “General” tab. You’ll see that the *Work* field shows 520 hours (65 days times 8 hours per day). Change the 520 hours to 65 hours to reflect just one hour of management oversight per day. What you have now told *ProjectLibre* is that over a duration of 65 days this task will require 65 hours. The Gantt chart should not have changed since the task is still 65 days long. If you change either the *Duration* or the *Work* fields, the software will try to do what it thinks you want to have done. Because of how easy it is for the user to be thinking one thing and the software interprets it in a different way, Fixed Duration tasks can take a while to get used to. For example, if you change the *Duration* to be longer, the *Work* will increase in the same percentage. (The software is presuming that the same rate of having to do this task will continue now for a longer period.) If you shorten the *Duration*, however, the *Work* can stay the same on the presumption that the work that needed to be done still needs to be done but now has a shorter amount of time.

The other option that works better for an ongoing task is to change the task’s type to “Fixed Work” in the task description’s “Advance” tab. This now tells *ProjectLibre* that the ratio of *Work* to *Duration* is a constant no matter how short or long this task is. Now, having *Duration* be 65 days and *Work* be 65 hours does exactly what we want: The software knows that this task takes one hour per day and if we extend the *Duration* to 85 days, for example (and will be longer after all the resource issues are resolved), the *Work* automatically grows to 85 hours.

Note that when you assign a person to a task of either type “Fixed Duration” or “Fixed Work”, the software will automatically determine the percentage of this person’s time that is required to fulfill the task.

Schedule and Costs

Once the initial resources are allocated, the next step is to verify that the right people are doing the right tasks. To illustrate this, the exercise of allocating resources in the previous section was continued to the end of the project. The following people assigned to these areas throughout the project: TM1 provides management oversight and codes function C, TM2 does the marketing, TM3 is a programmer given functions A and B, TM4 is the designer and codes function D, TM5 designs the tests, TM6 conducts the tests, and TM7 writes the user documentation. Over-commitments were resolved by adding delays. The Gantt chart prior to starting the project now looks like this:



As an aside, several steps ago (page 20/21) when we first adjusted TM2’s *Units* on the Marketing Study (Task 2) to 50%, the schedule increased in length quite a bit and this question was asked: “Is this new Task 2 completion time a problem? We won’t know until we assign the resources to the rest of the tasks.” Now we can determine this. After the other team member’s work schedule were factored in, the whole schedule has increased enough that this Marketing Study task is no longer the only item on the critical path. (Designing Round 2 cannot begin until some of the Round 1 testing has completed as well.)

Now that the project tasks are allocated to resources (people) and their workload has been leveled, we can now determine the cost of the project. (The reason this should not be done before leveling the workload for the team members is that project-long costs, such as the management overhead, cannot be accurately determined without knowing when the project will end.) To see a report showing how much this project will cost, click on View | Other Views | Report. Towards the top middle section of the screen there will be a menu selector entitled “Report:”. Selecting “Project Details” results in this output:

tutorial version two

Dates			
Start	8/4/14 8:00 AM	Finish	12/31/14 5:00 PM
Baseline Start		Baseline Finish	
Actual Start		Actual Finish	

Duration			
Scheduled	108 days	Remaining	108 days
Baseline	0 days	Actual	0 days
		Percent Complete	0%

Work			
Scheduled	1,271 hours	Remaining	1,271 hours
Baseline	0 hours	Actual	0 hours

Costs			
Scheduled	\$54,355.98	Remaining	\$54,355.98
Baseline	\$0.00	Actual	\$0.00
		Variance	\$0.00

This shows that the project is expected to take 108 normal work-days to complete, will finish at the very end of December, and will cost \$54,355 to complete. The Actual values are all zero as the software still has each task listed at 0% completed. (This cost is slightly low. Recall that when the team members were described TM1 was listed as coding at ½ the rate of the other programmers. Therefore, when TM1 is coding the *Units* has been set at 31% (the 75% maximum for this person minus 12% for the Management task divided by two) so that the schedule is correct. This does understate slightly how much TM1 will earn and thus understates how much the project will cost.)

Looking at the Gantt chart, we can see that TM1's coding of function C in all three phases is on the critical path. This is not good for two related reasons: Recalling the screen capture at the bottom of page 18 it shows that TM1 is tied with TM4 as earning the highest amount of anyone on the team yet TM1 programs only one-half as fast as the others. This is not a good combination for cost-effective use of the resources since the least expensive programmer, TM3, is not always busy. While it is easy to think that having more team members working on tasks would be better than having TM1 merely perform the management task, let's see what happens if all of TM1's programming tasks are assigned to TM3 with task delays added so that TM3 is not overloaded. Looking at the project report we see that the project will be completed faster and will cost less:

tutorial version two

Dates			
Start	8/4/14 8:00 AM	Finish	12/9/14 10:40 AM
Baseline Start		Baseline Finish	
Actual Start		Actual Finish	

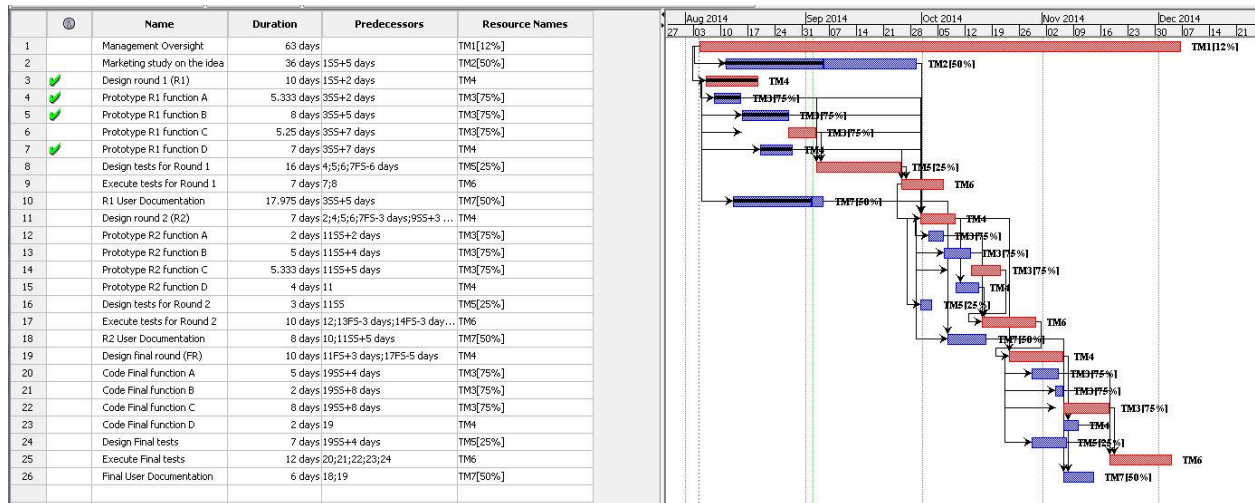
Duration			
Scheduled	91.333 days	Remaining	91.333 days
Baseline	0 days	Actual	0 days
		Percent Complete	0%

Work			
Scheduled	1,281 hours	Remaining	1,281 hours
Baseline	0 hours	Actual	0 hours

Costs			
Scheduled	\$52156.00	Remaining	\$52156.00
Baseline	\$0.00	Actual	\$0.00
		Variance	\$0.00

Please note that this is not to say that it is always cost ineffective to have management do programming tasks. I know many managers who are also very good cost-effective programmers. This example was created to show how using a project management tool like *ProjectLibre* can help you save time and money by looking at “what-if” scenarios.

Everything we have done so far has presumed that the project has not yet started. We will now presume that it is September 1, 2014 and the project has been going approximately according to schedule during the month of August. (The project manager can keep the tasks updated with the percent completion by clicking on the task and updating the percent completion field in the “general” tab periodically.) Setting statuses for this example, we will set that tasks 2, 4, 5, and 7 are complete, task 6 is slightly behind schedule and task 2 is slightly ahead of schedule. The Gantt chart now looks like the following (and note that the green vertical line shows where today is on the schedule):



If we look again at the summary financial data (View | Reports | Project Details), we see that the project has become slightly less costly due to the progress to date being ahead of schedule overall.

tutorial version two

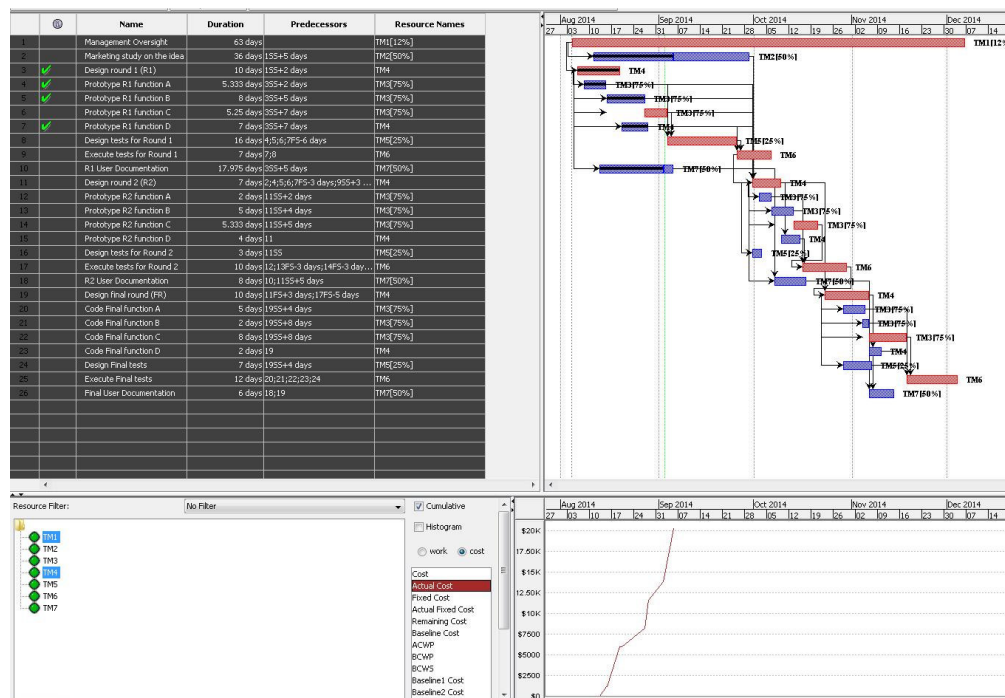
Dates			
Start	8/4/14 8:00 AM	Finish	12/9/14 10:40 AM
Baseline Start		Baseline Finish	
Actual Start	8/6/14 8:00 AM	Actual Finish	

Duration			
Scheduled	91.333 days	Remaining	91.333 days
Baseline	0 days	Actual	0 days
		Percent Complete	24%

Work			
Scheduled	1,280.5 hours	Remaining	907.7 hours
Baseline	0 hours	Actual	372.8 hours

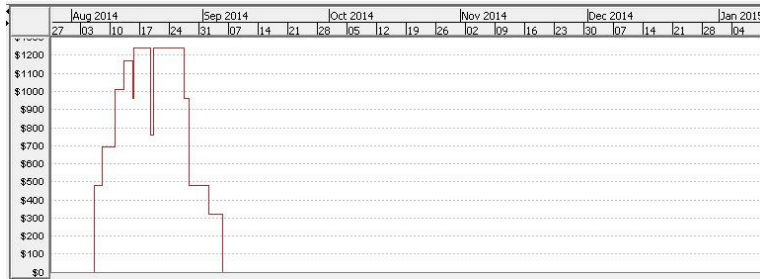
Costs			
Scheduled	\$52138.50	Remaining	\$34906.50
Baseline	\$0.00	Actual	\$17232.00
		Variance	(\$16080.00)

While this drop in estimated project cost is not dramatic (\$52,156 to \$52,138), maintaining approximately the original cost estimates like this fully one third of the way into the project is quite good news since it means that the estimates for how long each of the first round tasks will take was rather good. While this report shows the amount of remaining cost on the project, what is often also important is what the outgoing cash-flow will look like. In other words, not just *how much* remains to be spent, but also *when will it be spent?* If you recall where the button was to get the per-person workload histogram (the left-most button in the upper left), clicking on the button to its right will cause the charting features to appear. Towards the middle of the bottom section that just appeared you will see a list of chart types that can be requested. Many of these are advanced accounting principles, so we will concentrate on “Actual Cost” and “Cost” to help with cashflow analysis. If you click on “Cumulative” and select “Actual Cost” you will see nothing in the charting area. The reason is that the charting can either be for the entire project or can be done on individual tasks or group of tasks and we have not yet selected any tasks to chart. While keeping the charting section active, you can return to the Gantt chart as normal: Namely clicking on Task 1 Gantt. I find that seeing the cost of individual tasks is not particularly useful, so you can select all the tasks by clicking on the empty title box above the column of task numbers. Doing that will now show you the following:

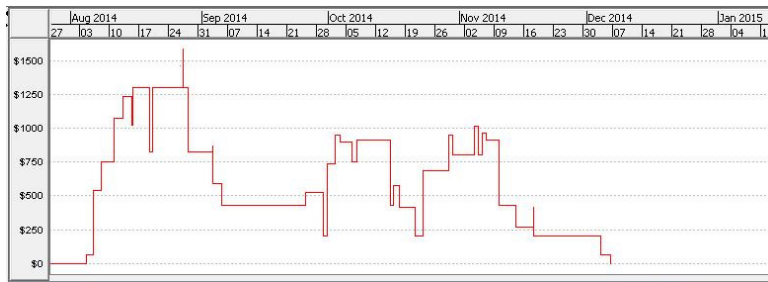


Note that the “Cumulative” button was selected thus this chart is showing the accumulated cost already spent on this project. (Please also note that for simplicity this sample project does not include costs like the fixed costs for the team members computer depreciation, office space, etc.)

Turning off the “Cumulative” button will show you the per-week cost of the project thus far:



If you now click on “Cost” instead of “Actual Cost”, ProjectLibre will show you the costs per period (which is weekly in the screenshots due to the zoom level) for the entire project and will now look like this:



(And no, I cannot explain the spike on Aug 26th that doesn’t show up on the Actual Cost chart above. Sorry.) Looking at this chart from the perspective of “today” supposedly being September 1st, we can see that the project’s costs should be reasonable during September, will go up in October and the first half of November and then will drop in mid-November as the final testing is completing. A real project, of course, would have developer time involved during the testing efforts for bug fixing, the costs for whatever distribution mechanism will be used, and supporting field test customers would have to be factored in. This example was obviously highly simplified so that it can concentrate on demonstrating ProjectLibre.

Good luck with your project management! As mentioned on Page 4, there are good resources available on Amazon for more details on project management in general and on ProjectLibre in particular.

Enjoy.

Appendix 1 (from V1): 2013 Team 3511 Task List

As mentioned above, Team 3511 used an Excel spreadsheet for the 2013 build to track their progress. Jim Broker of that team has graciously allowed the spreadsheet to be included in this Tutorial as it might help you break down the robot build into tasks. Since subsequent competitions won't be throwing Frisbees, your team's details will be different but this is a good example of a task-level that worked for a team.

